Secure and Resilient Cloud Computing for the Department of Defense

Nabil A. Schear, Patrick T. Cable, Robert K. Cunningham, Vijay N. Gadepally, Thomas M. Moyer, and Arkady B.Yerukhimovich

Cloud computing offers substantial benefits to its users: the ability to store and access massive amounts of data, on-demand delivery of computing services, the capability to widely share information, and the scalability of resource usage. Lincoln Laboratory is developing technology that will strengthen the security and resilience of cloud computing so that the Department of Defense can confidently deploy cloud services for its critical missions.

Imagine a military commander who urgently needs a specialized computing capability to analyze new intelligence, surveillance, and reconnaissance (ISR) data and integrate those data with existing ISR information. The commander directs his information technology (IT) staff and developers to design this capability. The staff quickly provision computing hardware from a Department of Defense (DoD) cloud and compose the software and services needed to ingest, enrich, create, and share knowledge from the data, while ensuring that the resulting capability remains secure and resilient (i.e., able to continue operations after a disruption). Within days, the staff has an initial system for analyzing the ISR data up and running. In the following weeks, they enhance the system by creating new features and adding capacity for even more data. This vision for agile, inexpensive cloud computing could revolutionize the way the DoD operates, and Lincoln Laboratory is building the next-generation secure cloud computing systems that could enable that vision.

Marketers have made the term cloud synonymous with ubiquitous, convenient computing. Digging below this simplified description, we find that cloud computing is a model for deploying software and hardware resources at lower cost and with greater flexibility than deploying typical enterprise computing resources. The defining attributes of cloud computing include on-demand self-service, broad network access, resource pooling, rapid elasticity (i.e., ability to adapt quickly to changing computational demands), and measured service (i.e., accounting and billing of resource usage) [1]. In cloud computing, computation and software capabilities are outsourced to a provider that delivers services to a cloud user (also called a tenant).

The DoD is looking to the cloud computing model as a means for lowering the costs and improving the flexibility of computing systems while delivering more capable services. But, the process of moving to the cloud is not without peril. The 2013 Defense Science Board's Report of the Task Force on Cyber Security and Reliability in a Digital Cloud recommended that "DoD should pursue private cloud computing to enhance mission capabilities, provided that strong security measures are in place" [2]. The study team, including experts from Lincoln Laboratory, the DoD, commercial cloud providers (e.g., Google and Amazon), and leading universities, found shortcomings in the security and resilience of clouds. The study further highlighted the need for research addressing conditions of interest to a warfighter, whose computing resources may face an active cyber adversary, intermittent connectivity, and physical attacks on computing hardware.

Today's cloud providers and the technology that underpins them are focused on the availability and scalability of services and not on DoD-specific security needs. Commercial

Distribution A: Public Release. This work is sponsored by the Assistant Secretary of Defense for Research & Engineering under Air Force Contract #FA8721-05-C-0002.
Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government.

cloud security is typically proprietary and thus opaque to tenants. For example, tenants have no visibility into cloud network security or data access. The prevailing cloud computing model is based on users trusting their cloud providers; data are stored unencrypted inside the cloud and all processing is done on unprotected data. The only enforceable guarantees that tenants have are through legal service-level agreements that loosely define the security responsibilities of both providers and users. This legal model does not provide tenants with timely and controllable mechanisms with which to respond when adversaries strike. As the DoD seeks to utilize commercial cloud technology, current cloud security will leave the DoD unable to protect their cloud resources from external attack, their cloud provider, insiders, or malicious tenants.

To address these shortcomings in cloud security, Lincoln Laboratory has undertaken the Lincoln Laboratory Secure and Resilient Cloud (LLSRC) effort to shore up the technology behind the cloud. The LLSRC approach is to (1) define a more accurate threat model for DoD cloud computing, (2) research and build technology that addresses that threat model, and (3) integrate the technology into a usable, secure, resilient cloud test bed. Underpinning this work is the semitrusted cloud threat model, which is built on the assumption that some of the cloud infrastructure or resources will be under the control of an adversary, but that there remains a portion of the cloud that can be inspected and trusted.

Our research and prototyping efforts are focused on four key components needed for a secure and resilient cloud: communication, storage, processing, and a high-assurance architecture that holds them together. In each area, our goal is to achieve security and resilience in the semitrusted cloud threat model. The vision for this technology is to create an ecosystem of services and capabilities that allows the DoD to build secure, resilient cloud mission applications. We are developing services and interfaces that can be recomposed to meet mission needs. Finally, we are combining these prototypes and services in a cloud test bed that reduces the risks for the DoD's acquisition of secure, resilient cloud technology by providing proofs of concept, technology maturity, integration demonstrations, and security evaluations.

Cloud: A Primer

Cloud computing changes how information services can be created and implemented. Before the cloud era, providing a new computing service (e.g., a large website or a file server) meant substantial capital expenses for data center space, network connectivity, and servers. After the capital investment, companies needed large teams of IT personnel and developers to manually build, install, configure, and maintain the supporting infrastructure. It took months for the teams to field the new service and considerable expense to operate and maintain it. In the cloud model, computing resources can be created on demand and composed into applications quickly.

The National Institute of Standards and Technology (NIST) defines the cloud as a "model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [1]. NIST's five essential characteristics of a cloud service are contrasted with those from the "old way" of enterprise computing in Table 1.

Table 1. Cloud versus Enterprise Computing Characteristics

Cloud Computing	Enterprise Computing
On-demand self-service	User request and long implementation to provide services
Broad network access	Limited to local area/company networks only
Resource pooling	Dedicated resources; expensive resilient hardware
Rapid elasticity	Fixed, over-provisioned capacity; expensive to scale up/down
Measured service	Poor metrics; unmeasurable guarantees

Cloud offerings fit into three different service models—infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS)—that target system administrators, developers, and end-users respectively (see Table 2). NIST identifies four environments in which cloud services exist: public, community, private, and hybrid. Public clouds allow anyone to use them. Community clouds may share tenants across a particular community of users (e.g., a cloud managed by the Defense Information Systems Agency [DISA]). Private clouds are generally limited within an organization (and their broad network access may be more limited). Hybrid clouds allow the mixing of private and/or community clouds with public ones. Some organizations have a private cloud for general use but may need to scale to broader cloud services for specific needs during certain periods.

Table 2. Cloud Service Models

Service Model	What's Provided	Flexibility	Examples
IaaS	Compute, storage, and network	High	Amazon Elastic Compute Cloud (EC2), DISA milCloud, Google Compute Engine, Microsoft Azure
PaaS	Application program interfaces (API) and services	Medium	Amazon Elastic MapReduce, MathWorks Cloud, Red Hat OpenShift
SaaS	Full-fledged applications	Low	Google gMail, Microsoft Office365, Facebook

When building cloud applications for these service models and environments, developers and designers must consider distributed systems issues, such as consistency, availability, and network failure. Similarly, cloud developers and system administrators need to resolve how to automate the deployment and monitor the availability of an elastic distributed system. These issues represent a considerable departure from the vast majority of enterprise computing patterns. Indeed, many DoD cloud initiatives thus far have not broken the mold of enterprising computing; they are simply performing virtualization at a distant data center. Solving the challenges presented by distributed systems will benefit the future of DoD software by providing increased resiliency to the end-users' missions.

Semitrusted Cloud Threat Model

Cloud services are attractive options for computing. They are easy to create, are usually straightforward to use, and offer flexibility and low cost; however, they carry significant security risks. Consider the example of Code Spaces [3]. In June 2014, attackers stole the credentials to the company's Amazon Web Services cloud account and proceeded to destroy all Code Spaces' virtual machines and customer data. Unable to recover, the company ceased operations shortly thereafter.

The first steps toward combating threats to the cloud are to understand and to codify assumptions about attacks and risk by examining the prevailing threat model. The dominant commercial cloud threat model is based on trusting the cloud providers and their system administrators. The layered service model of infrastructure, platform, and software as services allows a buyer of cloud services to abstract away the details of the lower layers. This model often results in security that is opaque to the end-user. For a number of reasons, security opacity is beneficial to providers. First, the providers' infrastructure and associated mechanisms for the security of their offerings are their critical intellectual property that the providers are not incentivized to share. Second, by not specifying the details of their security implementations, providers are free to change the details as needed without violating any service-level agreement. Last, providers can espouse a shared security responsibility model in which attacks and vulnerabilities occurring at or below the level of the providers' services are the providers' responsibility, and attacks and vulnerabilities above the providers' services are the responsibility of the cloud users. Rarely are sophisticated real-world attacks so cleanly separated across the layers of a computing stack; therefore, cloud providers can indemnify themselves of liability and blame the users for any security breaches that arise.

A further problem with the prevailing security model for clouds is that of mismatched priorities and control. Since cloud service providers require their users to outsource security to the providers, users must also give up control of how to respond to an attack, thereby allowing providers to both prioritize and formulate the responses. With only vague security service-level agreements in place as leverage, cloud users are at the providers' mercy when an attack happens.

An alternate threat model is one in which cloud providers are not trusted at all because they are a third party to users' resources. This conservative approach is taken by some users in the DoD who are engaged in sensitive missions and also by many academic researchers , especially those working in cryptography. The assumption that the cloud is completely insecure leads to the use of very inflexible solutions (e.g., encrypting data and not processing them at all in the cloud) or extremely expensive operations (e.g., using fully homomorphic encryption that performs computation without decrypting data [4]). As a result of confining technology to this threat model, many of the benefits of cloud computing are lost. Furthermore, a conservative user may avoid using the cloud at all and fall back to single-tenant enterprise computing.

We choose neither of these extremes for the LLSRC threat model. Instead, we use a trust model that we call the semitrusted cloud threat model (see Figure 1). In the semitrusted model, we assume that some fraction of the cloud resources is under the control of adversaries. We neither distinguish between external or insider attackers nor assume that we can always precisely identify which nodes are corrupted and which are unaffected. Applying this threat model, we expect and design for cloud compromise, but we assume that some trustworthy base of resources remains on which we can build secure and resilient systems.

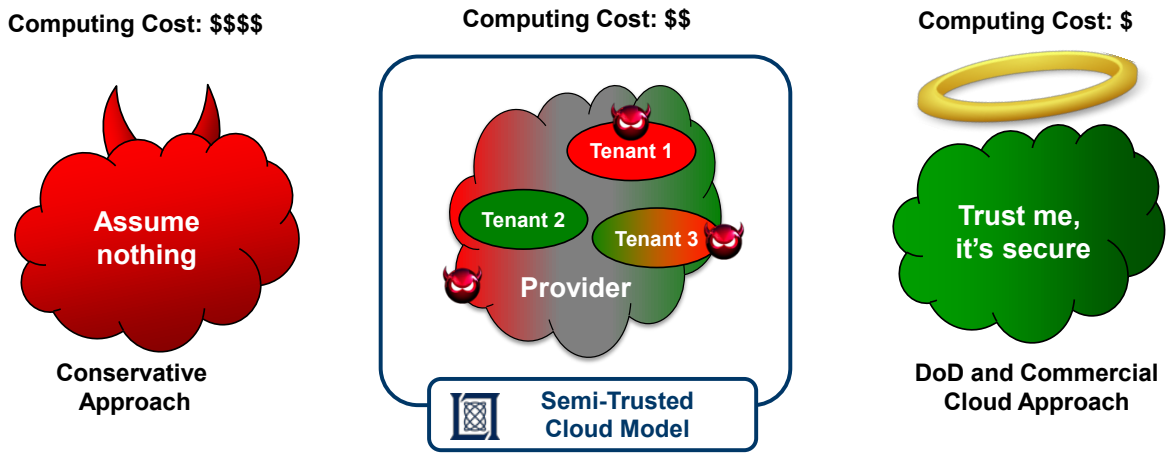

FIGURE 1. This comparison of cloud threat models shows how the semitrusted threat model Lincoln Laboratory advocates accepts a reasonable amount of risk to maintain the cloud computing benefits of low computational costs and flexible, scalable services.

Secure and Resilient Cloud Architecture

Building a secure and resilient cloud system for the DoD necessitates some changes and additions to the standard cloud architecture. The LLSRC architecture stack (Figure 2) begins at the bottom with the same commodity, low-cost hardware present in today's clouds but with the addition a hardware root of trust, (i.e., a specialized cryptographic coprocessor, that is trusted by the operating system). Atop that layer is a high-assurance trusted computing architecture that allows us to bootstrap (initiate) trusted cryptographic keys that will underpin higher layers. Because this layer also supports bidirectional control and visibility of the cloud infrastructure below, cloud tenants can obtain actionable situational awareness of the resources they are using. Above that trusted architecture, we build systems that enable the three core capabilities needed for cloud computing: communication, storage, and processing. We aim to develop systems that maintain security and resilience in the face of adversaries who control some of the cloud resources.

The LLSRC architecture fits both beneath and alongside existing insecure cloud software. As a result, we need a strategy for integrating LLSRC technology with the cloud services and applications that need to be secured. The LLSRC integration strategy is to utilize a suite of services and tools at various levels of the cloud stack. This strategy provides a tiered approach to integration that starts with limited-invasiveness, compatible solutions that can be deployed transparently to the applications. The next integration point is the security application program interfaces (API) and services that developers can compose. The final integration point consists of full replacements for end-user software. These integrations plug in roughly at the infrastructure, platform, and software layers to afford cloud tenants the maximum flexibility to design and compose appropriate solutions for their missions' needs.

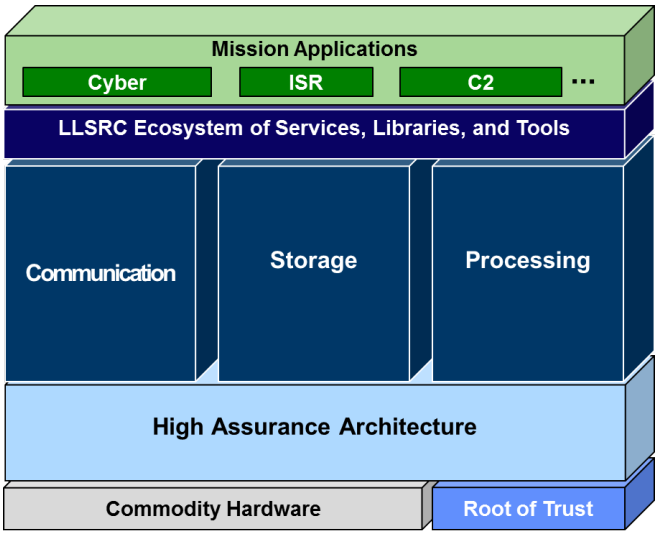

FIGURE 2. At the base of the stack for the Lincoln Laboratory Secure and Resilient Cloud (LLSRC) are commercially available hardware and a root of trust. The high-assurance architecture allows the three core capabilities to take advantage of its security measures. The LLSRC application interfaces and the applications at the higher layers allow developers to design software for specific missions, such as intelligence, surveillance, and reconnaissance (ISR) or command and control (C2).

High-Assurance Architecture

Today's cloud service providers do not furnish the building blocks necessary to establish a trusted environment for hosting mission-critical applications and data. Tenants have limited ability to verify the underlying platform when they deploy their software and data to the cloud provider and to ensure that the platform remains in a good state for the duration of their computation. Additionally, current practices restrict tenants' ability to establish unique, unforgeable identities for individual nodes that are tied to a hardware root of trust. Often, identity is based solely on a software-based cryptographic solution or unverified trust in the provider. What is needed are mechanisms to establish trusted cloud identities, rooted in hardware, and to maintain appropriate situational awareness and control over cloud nodes.

To establish a cryptographic node identity with a hardware root of trust, nodes validate their environment and, in turn, are validated by the environment before being issued a long-term identity. The hardware that is often used to establish a trusted environment in commodity systems is the Trusted Platform Module (TPM), a small cryptographic processor that provides a hardware root of trust. Figure 3 shows the process by which a system with a TPM will boot, known as secure boot. Secure boot ensures that no files have been modified and halts the system when any unexpected changes are detected. The basic input/output system (BIOS) starts the process by measuring (or validating) the firmware within the system. The firmware then validates the boot loader, which in turn validates the operating system (OS). The OS then monitors the applications running on the system.

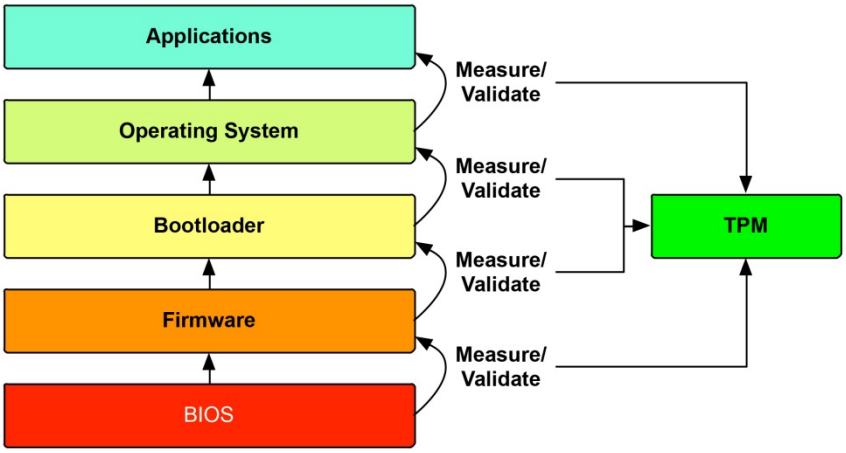

FIGURE 3. A trusted computing architecture validates the current environment during system boot to allow the system to generate proofs that show the current integrity state of the system. The validation is enabled by the Trusted Platform Module (TPM) and sequences through all layers of the stack.

During normal operation, these collected measurements can be provided to remote systems as a means of proving, or attesting, the system's integrity state, a process known as integrity measurement. The TPM forms the hardware root of trust for secure boot and integrity measurement. This root of trust is expected to function correctly, and from this assumption, we can validate the entire set of applications running on the system, using the chained validation described in the preceding paragraph and Figure 3. These techniques ensure that any deviation from a known-good state can be detected so that appropriate responses can then be taken. This chained validation approach increases the difficulty for an adversary who is attempting to compromise a system while avoiding detection.

The first challenge to extending trusted computing to the cloud is virtualization. Because virtual machines are by definition separated from and unaware of the underlying hardware on which they run, we need a way to tie the virtual environment to one rooted in hardware. The software Virtual Trusted Platform Module (vTPM) solves this problem by linking its attestations of the virtual environment to that of the underlying hardware (e.g., the hardware of the virtual machine monitor, or hypervisor) [5].

The second challenge to trusted computing is effectively scaling the techniques to the thousands of nodes in the cloud. The sheer number of machines and limitations of the performance of hardware TPMs (e.g., a single digital signature, which is a fundamental building block in trusted computing, can take ~1 second to produce) make it infeasible to have each virtual machine attest to all other hosts with which it communicates directly. We propose using a cloud verifier to alleviate this problem by centralizing integrity measurement to a dedicated software service that verifies all the nodes belonging to a particular entity (tenant or provider) [6]. This integrity measurement asynchronously occurs separately from the communication channel and has no impact on the performance of the application's communication. While centralizing the integrity measurement reduces the timeliness of detecting violations of a system's integrity, the number of systems that can be attested is more scalable.

Using hardware-rooted node identities and the measurements collected by their individual cloud verifiers, tenants can now create and maintain their own individualized

trusted membership lists. The long-term identities (i.e., public/private key pairs) in the trusted membership list bootstrap both long-term identity and ephemeral keys (temporary keys generated for each key establishment process) for higher-level services. Changes to a trusted membership list inform higher-level services of the trust level of each node in tenants' environments. The trusted membership list forms the foundation and interface that enables secure communication, information flow tracking, and storage.

To demonstrate this, we created Keylime, a collection of services and an API that allows applications on a cloud node to bootstrap trust in the manner described. Keylime provides the cloud verifier (CV) service and the cloud node service that runs on each compute node. The ability to maintain a trusted list of cloud compute nodes depends on Keylime's ability to continuously verify TPM quotes. To test how scalable our library was, we created an automated test platform to create new virtual "compute nodes" and verify their quotes. Figure 4 shows that we were able to reach a rate of 2500 verifications per second. This demonstrates that our system can scale to handle thousands of nodes with low latency detection of integrity violations.

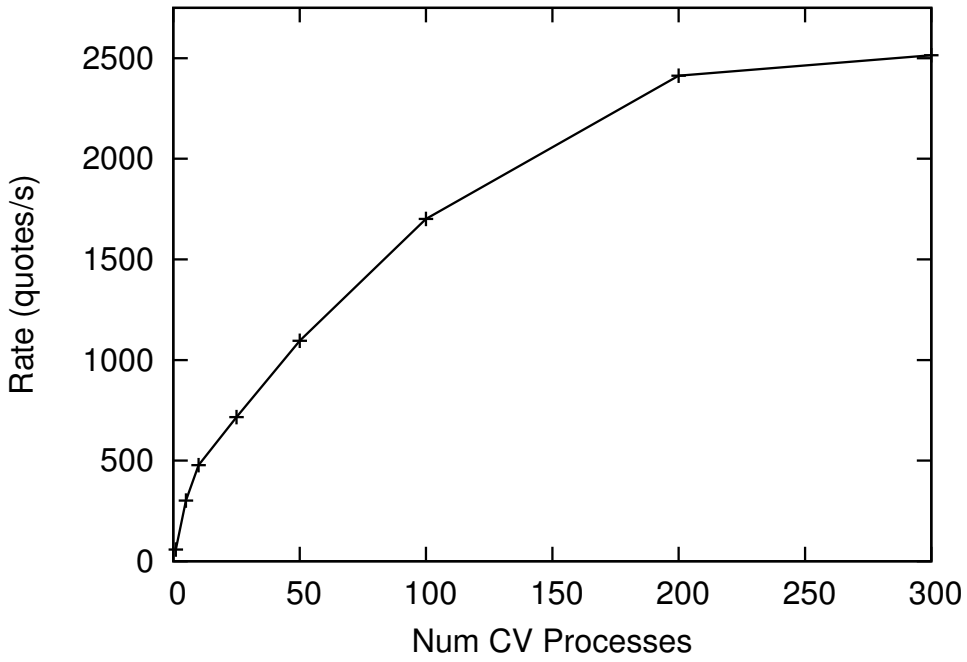

Figure 4. Measuring the scalability of the Keylime TPM verification library

Secure Communication

Cloud providers, malicious insiders, or other tenants mounting a side-channel attack[1] may eavesdrop on cloud networks. To protect data in transit, we can rely on well-known techniques, such as Internet protocol security (IPsec) or transport layer security (TLS), which can provide both confidentiality and integrity of network traffic. In LLSRC, we use the long-term identity keys provided by the architecture to bootstrap the creation of ephemeral cryptographic keys for use with IPsec. By using this technique, we can secure communications without presharing keys to all nodes, and we do not need each node to be provisioned with the same keys. Each system has an agent that monitors changes to the

[1] A side-channel attack is one that exploits nontextual information, such as timing information or power consumption statistics, generated by an encryption device).

trusted measurement list and can terminate IPsec tunnels when nodes are removed. This solution can be preconfigured as part of the cloud offering and transparently protects all inter-tenant cloud communications to provide an easy mechanism by which all point-to-point cloud network traffic can be encrypted. However, one limitation of this approach is the need for point-to-multipoint traffic.

Brokered publish/subscribe services offer a way to broadcast messages to multiple nodes. When nodes first join the service, they specify to a broker a list of topics in which they are interested. When nodes send broadcast messages related to those topics to the broker, the broker sends the messages to all the nodes that have subscribed to that topic. To enhance scalability and elasticity, tenants can leverage a broker operated by the cloud provider to multiplex multiple tenants' messages across the cloud efficiently [7]. While this practice has many advantages, the broker presents adversaries or malicious cloud insiders with an easy target to attack.

To address the insecurity of brokered cloud messaging systems, we are building a cryptographic overlay that protects data passing over an untrusted broker. The system works by running a proxy on each cloud node that either publishes or receives messages from the cloud broker. These proxies use dynamic group keying to establish and distribute a cryptographic key that tenants can control for each topic [8]. The proxies then encrypt all data that transit the broker and subsequently remove any encryption before delivering the data to the destination application. The trusted membership list provides the keys needed to securely distribute the topic keys and the cues for when to rekey as the membership list changes. This solution is transparent to existing cloud applications (requiring only a configuration change to redirect the application to the proxy rather than to the real broker), and it maintains the elasticity and scalability of the cloud broker.

Secure Cloud Storage

The ability to store and access data securely is core to developing a protected cloud infrastructure. Threats to data storage security abound in the cloud. Examples include insiders maliciously accessing physical disks, malware modifying critical files, providers improperly sanitizing reused media, and services providing insufficient access control. Again, we turn to cryptography and key management to simplify the trust assumptions we must make to ensure critical data are protected.

The simplest storage media to protect in the cloud is the local storage attached directly to a cloud compute node. While commercial products from HyTrust and SafeNet exist to transparently encrypt these volumes, these products fall short because they rely on software-based or password-based trust with no linkage to a hardware root of trust or to the integrity state of the system. Using the LLSRC high-assurance architecture, we can mitigate this shortcoming. As with the communication component of cloud computing, long-term identity keys can be used to unlock cryptographic keys that protect local storage, and agents monitoring the tenant's trusted membership list can revoke access to encrypted volumes as the cloud node integrity state changes. By leveraging hardware acceleration and the bootstrapping capabilities of the trusted membership list, this solution can be deployed transparently to applications with minimal performance overhead.

Often, local cloud node storage is not used to store persistent state because nodes may come and go as the storage load varies. When persistent state is not stored in a cloud node, that state must be stored elsewhere and is typically shared with other cloud nodes. Object storage systems (e.g., Amazon Simple Storage Service (S3), OpenStack Swift) or distributed file systems (e.g., Lustre, Ceph) often fill this need. These systems typically rely on access

control lists that are enforced by the system. For example, file systems compliant with POSIX (a set of standards known as Portable Operating System Interface for UNIX) will offer owner, group, and other read-and-write permissions.

To move this reference monitor model of access control to one based in cryptography, we need both dynamic group keying and long-term identity keys provided by the architecture. Lincoln Laboratory has developed a prototype system that encrypts data in an object storage system and mediates access to shared resources using key management [9]. Using a method similar to the protection of topic keys with a secure publish/subscribe proxy, our system creates a randomly generated key, called the content key, for each object (i.e., piece of data) in the system and encrypts the object using the content key. The system then encrypts the content key using the long-term identity key of each entity that has permission to read the object. These encrypted keys, or key wraps, along with the encrypted object, are stored in the object storage system. The owner of the object can add permissions later and can revoke access by re-encrypting the object under a new key or encrypting under a new key only when new data are written to the object. An asynchronous agent running in the cloud manages when permissions should be updated based on changes to the membership list.

The final and most complex storage application to secure is that of databases supporting complex queries. Databases are critical to cloud computing because they allow applications to efficiently access small portions of large datasets. The standard sets of cryptographic algorithms (e.g., block ciphers, cryptographic hash functions, and public key cryptography) are insufficient for use with databases because these algorithms do not allow search operations on ciphertext. We need new cryptographic techniques, such as deterministic, order-preserving, and searchable encryption [10–12]. These cryptographic techniques have proven to be both secure and practical for relational or SQL-style[2] databases. These protections can be deployed with approximately a tenfold decrease in processing overhead compared to the overhead involved in plaintext operations for a wide variety of advanced queries, including substring matching and ranges [13, 14].

Many cloud applications are moving to a schema and storage pattern that relaxes some of the constraints of SQL to create massively scalable databases. To address this shift, we have developed the Computing on Masked Data (CMD) toolbox, which employs the aforementioned encryption primitives to provide a high-performance framework for securing data in NoSQL databases like Apache Accumulo [15, 16]. Specifically, to allow users to securely mask data stored in a database, the CMD toolbox makes use of encryption techniques such as semantically secure encryption (RND, which only supports data retrieval), deterministic encryption (DET, which supports database matching queries), and order-preserving encryption (OPE, which supports database range and match operations).

When data are inserted into the cloud database, they are first converted to sparse representations known as associative arrays (sparse matrices with text-labeled rows, columns, and values). Prior to inserting data into the cloud, users select the appropriate level of masking (i.e., encryption with a secret key) that they will apply to support the security and functionality goals of their application.

To retrieve the masked data, users submit a masked query (with the same key used when they inserted data) or an analytic (such as correlation or thresholding). To view the results, users unmask their data by using the same key they used to mask the data. The

[2] SQL, short for Structured Query Language, is a widely used programing language for managing databases.

CMD prototype, which has been applied to health-care data, network logs, and social media datasets, adds no more than twice the relative computational overhead. For example, Figure 5 describes the computational time taken to perform a correlation on masked data (DET and OPE) and plaintext data (CLR). We found the time taken to perform masked correlation (represented by the red OPE line and blue DET line) is within a factor or two of performing the correlation on plaintext data (represented by the black CLR line).

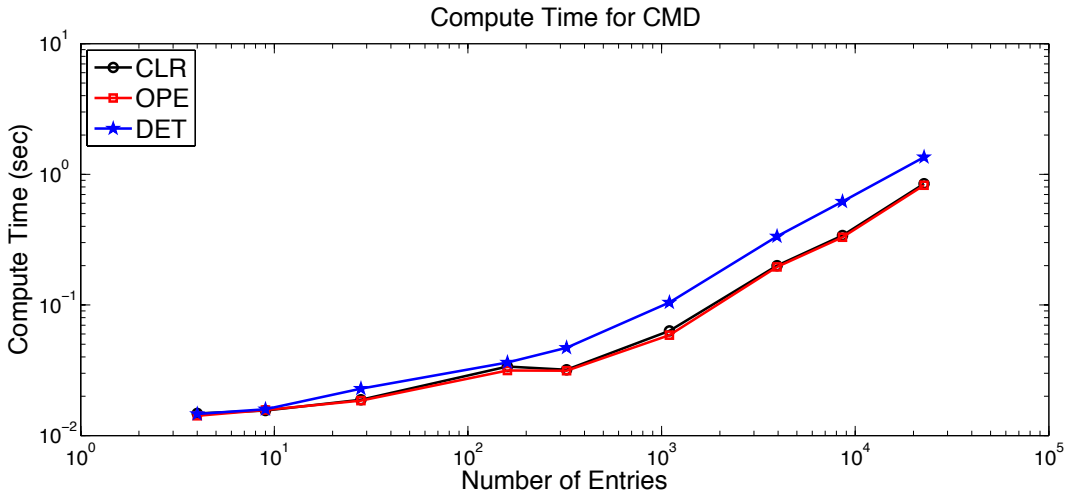

Figure 5. Time to perform a masked computation (red and blue lines) compared to plaintext (black line).

To address the integrity of data stored in cloud databases, we are developing a system to automatically and transparently append digital signatures to all data items in the NoSQL Accumulo database. In addition to developing techniques for protecting data integrity, we are developing a system that uses tools like Merkle hash trees to safeguard the soundness of results returned by the database [17, 18]. This system will ensure that a database cannot suppress or falsify results without detection. Both these systems are implemented in the software the client uses to access the database. If database clients reside in other cloud nodes (that are powering other applications), we can leverage the identity keys from the trusted membership list to authenticate nodes and distribute keys for signing and verifying data in the database.

Secure Processing

The most challenging problem in cloud security is the protection of data while they are being processed. Since it may often be desirable for users to perform computations, such as statistical analyses, over data stored in the cloud, it is important for cloud providers to ensure that the data are secured throughout the computation. However, most clouds today only allow processing over unencrypted data to give the cloud the access necessary to perform the computation. This approach requires data owners to give up control over their data and trust the cloud to do the right thing. If even a single one of a tenant's cloud nodes is compromised by an adversary, either through malware or the presence of a malicious

insider, then both the confidentiality of the data and the integrity of the computed results may be compromised. Although the trusted computing techniques we have described can detect when the system reaches an unknown or malicious state, these techniques still leave the data vulnerable as they are being processed and stored in memory on a cloud system.

Consider the proposed semitrusted cloud threat model presented earlier. In this model, we explicitly assumed that some fraction of cloud machines is under adversarial control at any given time. Under this threat model, we cannot rely on trusting the entire cloud, and additional protections are necessary to keep data confidential during processing and to ensure correctness of computation results. A number of cryptographic techniques, such as homomorphic encryption, verifiable computation, and multiparty computation (see [19] for a brief survey), have been proposed by the academic community to protect data during processing in a semitrusted computing environment; however, more research is necessary for developers to understand the applicability and practicality of these techniques in a cloud setting.

To achieve secure processing on a semitrusted cloud, we are investigating the feasibility of secure multiparty computation (MPC) in the cloud. Secure MPC allows multiple parties to work together to compute a joint function on their private data without revealing those data to each other or any external parties and while ensuring that correct results are obtained even if a small number of the parties misbehave. The MPC arrangement is an effective substitute for a scenario in which a perfectly trusted third party would be needed (Figure 6). Distributed computation like MPC arises quite naturally in a cloud setting where data may be distributed over multiple cloud nodes or may belong to different cloud tenants. Using MPC, cloud tenants can perform computation over distributed sensitive data while protecting the confidentiality and integrity of the input data and the results, even if some fraction of the machines involved in the computation is corrupted. In fact, this security guarantee does not even require that the identities of corrupted parties be known.

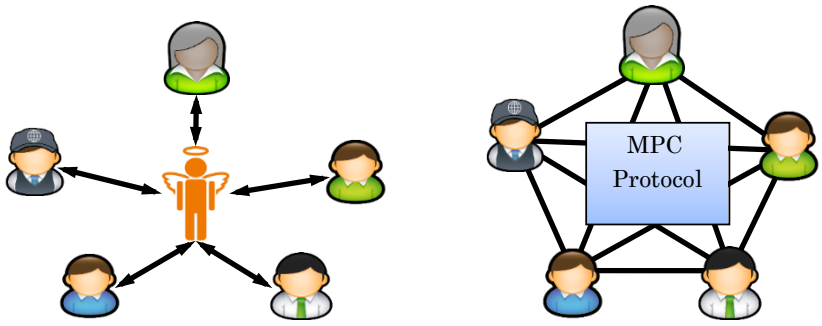

FIGURE 6. Multiparty computation (MPC) on the right emulates a trusted third-party scenario (on left) to achieve comparable security.

The academic cryptography community originally developed secure MPC in the mid-1980s [20–23]. However, for more than 20 years, the computing community considered MPC protocols purely theoretical novelties and explored few, if any, real-world applications of the protocols. The perception that MPC is impracticable has been shattered in the past few years as multiple efficient protocols have been implemented and used for real-world applications such as secure auctions [24] and private statistical analysis [25]. These

demonstrated uses of MPC protocols have confirmed that MPC is nearing readiness for real-world applications; however, much work remains before these protocols can be applied in a cloud setting. Specifically, it is necessary to build protocols that can perform efficiently for computations that are distributed over a large number of cloud nodes and that are optimized for the computations typically performed in cloud settings.

Lincoln Laboratory is currently addressing both of these requirements for cloud MPC. First, we are investigating ways to decrease the number of nodes that must communicate to each other in an MPC protocol. Reducing this communication locality is critical in a cloud network because the reduction lowers the total communication bandwidth necessary for MPC to run with many cloud nodes. Additionally, we are identifying and building MPC protocols for cloud-specific computations, such as shared cyber situational awareness and graph anomaly detection [26].

To better understand the usability and utility of MPC for such applications, we have developed an initial prototype of MPC for graph anomaly detection. Figure 7 shows the performance of both the unoptimized and optimized versions of our prototype. Our results demonstrate that several orders of magnitude improvements can be achieved by applying MPC-aware optimizations that transform existing algorithms into appropriate form for secure computation. Efficient MPC for such computations will demonstrate its applicability in a cloud environment and will open the path to adding strong security to cloud processing.

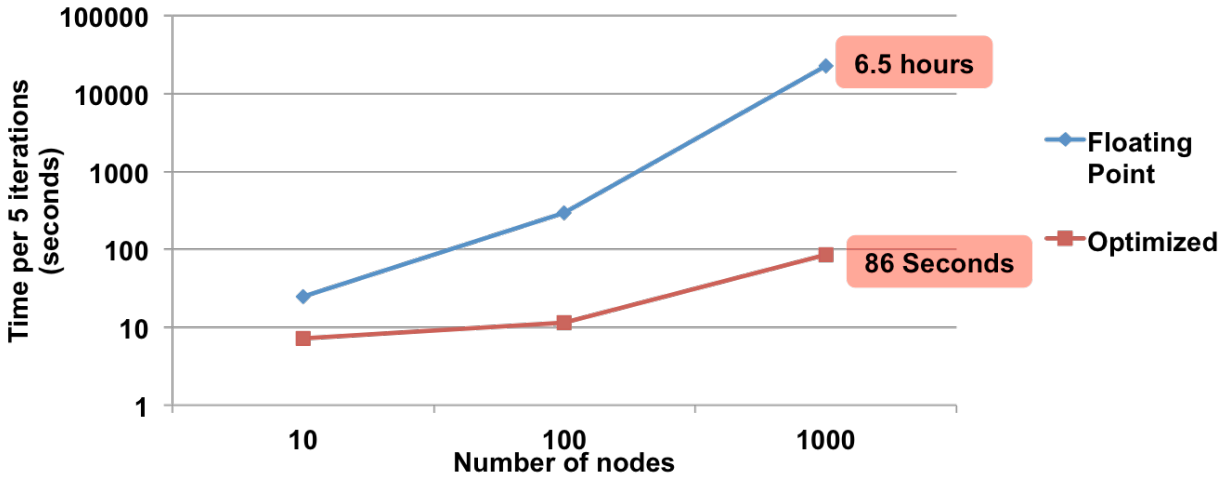

Figure 7: Running times of MPC for graph anomaly detection. The blue line represents an unoptimized initial prototype directly translating the algorithm into its MPC implementation. Whereas, the red line represents an MPC implementation of the same protocol using several MPC-specific optimizations (e.g., fixed-point arithmetic, sparse matrices, etc.) to significantly improve performance.

Achieving the LLSRC Vision

Developing the architecture and components discussed in this article represents a broad and sizable research agenda. Lincoln Laboratory is working on specific research and development (R&D) challenges within this space; however, we do not expect to solve all of them. We are leveraging promising R&D technology coming from academia, commercial companies, and other government labs and agencies. We are combining both our own technologies and those from other sources into an integration test bed.

Using the test bed as a platform for technology demonstration and evaluation, we hope

to evangelize the semitrusted cloud security model. By showing that technology *can* achieve stronger security guarantees than those achievable with the current trust models in which trust is full, unverified, and absolute, we may drive the adoption of technology that employs the semitrusted threat model, and we may influence changes in commercial offerings.

Acknowledgments

We appreciate the support of the following individuals working on and advising us on our work and this paper: Mayank Varia, Jeremy Kepner, Emily Shen, Eric Evans, Dave Martinez, Albert Reuther, Bryan Richard, Sasha Berkoff, Sophia Yakoubov, Ben Fuller, Andy Prout, Matt Hubbell, Roop Ganguly, Anna Simpson, Braden Hancock, Mike Calder, Chris Botaish, and Gene Itkis.

REFERENCES

1. P.M. Mell and T. Grance, "The NIST Definition of Cloud Computing," Technical Report SP 800-145, Gaithersburg, Md.: National Institute of Standards and Technology, 2011.
2. Defense Science Board, Report of the Task Force on Cyber Security and Reliability in a Digital Cloud. Washington, D.C.: Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics, 2013. Available at http://www.acq.osd. mil/dsb/reports/CyberCloud.pdf.
3. D. Goodin, "AWS Console Breach Leads to Demise of Service with 'Proven' Backup Plan," *ars technica*, June 2014. Available at http://arstechnica.com/security/2014/06/aws-console-breach-leads-to-demise-of-service-with-proven-backup-plan.
4. C. Gentry, "Fully Homomorphic Encryption Using Ideal Lattices," *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, 2009, pp. 169–178.
5. S. Berger, R. Cáceres, K.A. Goldman, R. Perez, R. Sailer, and L. van Doorn, "vTPM: Virtualizing the Trusted Platform Module," *Proceedings of the 15th Conference on USENIX Security Symposium,* vol.15, article 21, 2006.
6. J. Schiffman, T. Moyer, C. Shal, T. Jaeger, and P. McDaniel, "Justifying Integrity Using a Virtual Machine Verifier," *Proceedings of the 2009 Computer Security Applications Conference,* 2009*,* pp. 83–92.
7. Amazon Web Services, Simple Notification Service (SNS). Available at http://aws.amazon.com/sns/.
8. R. Khazan and D. Utin, "Lincoln Open Cryptographic Key Management Architecture," Tech Note, Lexington, Mass.: MIT Lincoln Laboratory, 2012. Available at http://www.ll.mit.edu/publications/technotes/TechNote_LOCKMA.pdf.
9. G. Itkis, V. Chandar, B. Fuller, J. Campbell, and R.K. Cunningham, "Rethinking Authentication: Iris Biometric Security Challenges and Solutions," *IEEE Signal Processing Magazine*, 2015 (accepted for publication).
10. M. Bellare, A. Boldyreva, and A. O'Neill, "Deterministic and Efficiently Searchable Encryption," chapter in *Advances in Cryptology—CRYPTO 2007*, Berlin, Heidelberg: Springer-Verlag, 2007, pp. 535–552.

11. A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill, "Order-Preserving Symmetric Encryption," *Proceedings of the 28th Annual International Conference on Advances in Cryptology: The Theory and Applications of Cryptographic Techniques*, pp. 224–241. Berlin, Heidelberg: Springer-Verlag, 2009.

12. R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions," *Proceedings of the 13th ACM Conference on Computer and Communications Security*, 2006, pp. 79–88.

13. M. Varia, B. Price, N. Hwang, A. Hamlin, J. Herzog, J. Poland, M. Reschly, S. Yakoubov, and R.K. Cunningham, "Automated Assessment of Secure Search Systems," *Operating Systems Review,* vol. 49, no. 1, 2015, pp. 22–30, published by the ACM Special Interest Group on Operating Systems.

14. R.A. Popa, C.M.S. Redfield, N. Zeldovich, and H. Balakrishnan, "CryptDB: Protecting Confidentiality with Encrypted Query Processing," *Proceedings of the 23rd ACM Symposium on Operating Systems Principles*, 2011, pp. 85–100.

15. J. Kepner, V. Gadepally, P. Michaleas, N. Schear, M. Varia, A. Yerukhimovich, and R.K. Cunningham, "Computing on Masked Data: A High Performance Method for Improving Big Data Veracity," *Proceedings of the 2014 High Performance Extreme Computing Conference*, 2014, pp. 1–6.

16. V. Gadepally, B. Hancock, B. Kaiser, J. Kepner, P. Michaleas, M. Varia, and A. Yerukhimovich, "Computing on Masked Data to Improve Big Data Security," *IEEE International Symposium on Technologies for Homeland Security*, 2015. Available at http://arxiv.org/abs/1504.01287

17. P.T. Devanbu, M. Gertz, C.U. Martel, and S.G. Stubblebine, "Authentic Third-Party Data Publication," *Fourteenth Annual Working Conference on Database Security: Data and Application Security, Development and Directions*, 2001, pp. 101–112.

18. M. Goodrich and R. Tamassia, "Efficient Authenticated Dictionaries with Skip Lists and Commutative Hashing," Technical Report, Johns Hopkins Information Security Institute, Baltimore: Johns Hopkins, 2001.

19. S. Yakoubov, V. Gadepally, N. Schear, E. Shen, and A. Yerukhimovich, "A Survey of Cryptographic Approaches to Securing Big-Data Analytics in the Cloud," *Proceedings of the 2014 High Performance Extreme Computing Conference*, 2014, pp. 1–6.

20. M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness Theorems for Non-cryptographic Fault-Tolerant Distributed Computation," *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, 1988, pp. 1–10.

21. D. Chaum, C. Crépeau, and I. Damgård, "Multiparty Unconditionally Secure Protocols," *Proceedings of the 20th Annual ACM Symposium on Theory of Computing,* 1988, pp. 11–19.

22. O. Goldreich, S. Micali, and A. Wigderson, "How to Play Any Mental Game or A Completeness Theorem for Protocols with Honest Majority," *Proceedings of the 19th Annual ACM Symposium on Theory of Computing,* 1987, pp. 218–229.

23. A.C. Yao, “How to Generate and Exchange Secrets,” *Proceedings of the 27th Annual Symposium on Foundations of Computer Science,* 1986, pp. 162–167.

24. P. Bogetoft, D.L. Christensen, I. Damgård, M. Geisler, T.P. Jakobsen, M. Krøigaard, J.D. Nielsen, J.B. Nielsen, K. Nielsen, J. Pagter, M. Schwartzbach, and T. Toft, “Secure Multiparty Computation Goes Live,” in *Financial Cryptography and Data Security,* Berlin, Heidelberg: Springer-Verlag, 2009, pp. 325–343.

25. D. Bogdanov, L. Kamm, S. Laur, and V. Sokk, “Rmind: A Tool for Cryptographically Secure Statistical Analysis,” International Association for Cryptologic Research, Cryptology ePrint Archive no. 2014-25525, 2014.

26. K.M. Carter, N. Idika, and W.W. Streilein, “Probabilistic Threat Propagation for Network Security,” *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 9, 2014, pp. 1394–1405.